COURSERA FINAL PROJECT

SPECIALIZED MODELS: TIME SERIES AND SURVIVAL ANALYSIS

Time Series Forecasting to predict grocery sales at Favorita stores

Project presented by

Raziel Amador Ríos, Ph.D.

to obtain the Coursera Certificate

Main object: produce a reliable forecast based on sales time-series data from a retail store (Favorita stores, an Ecuadorian-based company).

Barcelona, February 2023

Table of Contents

Time Series Forecasting

I

Main objective

The **main object** of the project is to:

- Generate a reliable **time series forecast** based on sales from Favorita stores.

Favorita stores is a grocery retail store based in Ecuador. Producing an accurate forecast could lead to decreased food waste related to overstocking and improve customer satisfaction. All the code can be found in the following GitHub repository, further information can be found in Supplementary information.

II

Data description

The dataset comes from **Kaggle**, named: Store Sales - Time Series Forecasting. The Kaggle dataset contains sales data from *Corporación Favorita*, a large Ecuadorian-based grocery retailer. Kaggle provides you with 7 diffeent files (see Table 1 for more details).

Number	File Name	Description
1	holiday_events.csv	Relevant holidays in Ecuador
2	oil.csv	Oil prices from 2013 to 2017
3	sample_submission.csv	submission example
4	stores.csv	Stores metadata
5	test.csv	Stores and family products
6	train.csv	Sales by store and product-family from 2013-01 to 2017-08
7	transactions.csv	Number of transactions by store

Table 1: Kaggle files description. Files are alphabetically sorted.

The training data represents 99% of the data, including dates from 2013-01-01 to 2017-08-16 (55.5 months), 54 stores placed in different cities within Ecuador, and 33 family-products (see Figure 1). The testing data includes dates from 2017-08-16 to 2017-08-31 (15 days).

Figure 1: Summary of the training dataset. The cluster information denotes similarity between stores.

III

Data exploration and data cleaning

Performing an exploratory data analysis (eda) of the sales from the training dataset, we can observe that grocery I, beverages, and produce are the top 3 most consumed products (see Figure 2) . Additionally, store type A, and cluster 5 are the most frequent among their classification (Figure 2).

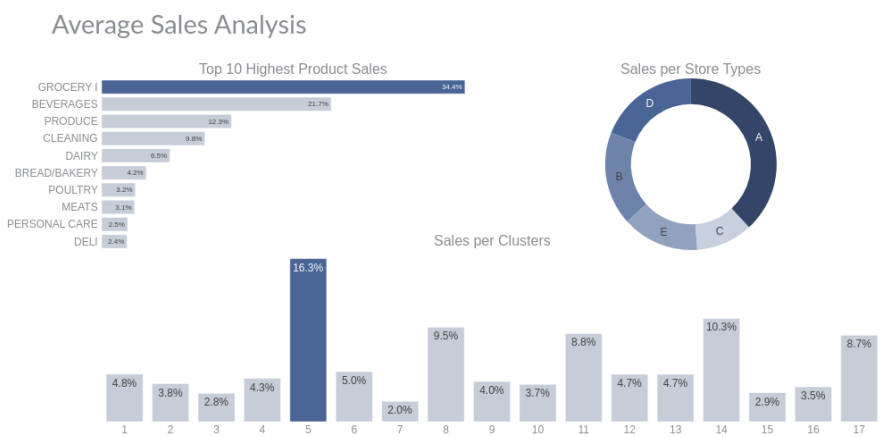

Figure 2: EDA of sales. The plot describes the sales by product, store type, and per cluster (left, right, and below, respectively). Darker blue represents higher sales.

The sales represented by Figure 3 shows low-peaks at the end of each year, which is explained because the stores are closed at New years. Moreover, we can observe a pattern at each year, with increased sales at the end of the year which overlaps with the Christmas eves, suggesting a seasonal pattern. These patterns are highlighted after smoothing the sales data with a 7 days moving average (the continuous red lines from the Figure 3).

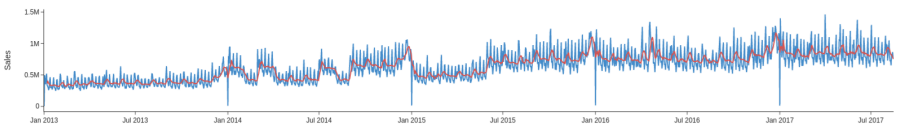

Figure 3: Target time series data. The y-axis represents the training sales from 2013-01-01 to 2017-08-15, and the x-axis shows the time in days (1,087 days). The sales were aggregated by all stores and products (see Figure 1). The raw data, and smoothed data (7 days moving average) are denoted by the continuous blue, and red lines, respectively.

III.1. Time series Decomposition

Time series are defined as a sequence of data organized in time order. The key components of time-series are: trend, seasonality, and residuals, displayed in Figure 4B, Figure 4C, and Figure 4D, respectively. Trend is the long-term direction of the time series, seasonality describes the periodic behavior such as holidays, and residuals represent the irregular fluctuations that we are not able to predict using the trend and/or seasonality.

In our target time series, we observe an upward trend, with a clear seasonality factor (this will be further analyzed below), and the residuals do not follow a random pattern. These insights will be used to select an adequate forecast model.

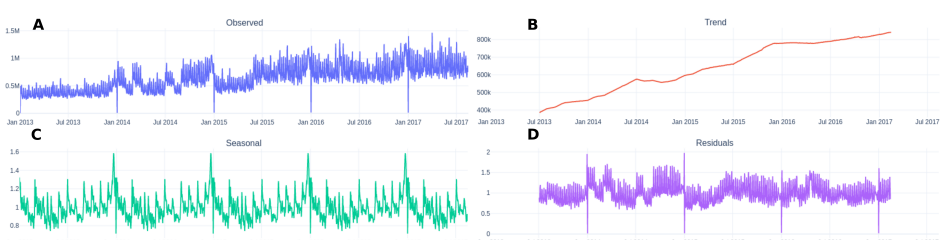

Figure 4: Time series decomposition. **(A)** Raw sales time series (sames as Figure 3). **(B)** Trend of sales. **(C)** Seasonality of sales. **(D)** Residuals of sales.

III.2. Stationarity and seasonality analyses

The stationary and seasonality are relevant components to select the adequate machine learning model (such as ARIMA, SARIMA, etc.) to generate a reliable forecast. A stationary time series is defined, if their statistical properties such as mean, and variance are all constant, and independent of time.

In consequence, we implemented the *Dickey-Fuller test* to assess stationarity. We obtained a *p-value* of 0.09, using a significance level of 0.05, we can reject the null hypothesis (the series is stationary) concluding that our series is non-stationarity.

In terms of seasonality, we aggregate the sales for each month and we can clearly see on average an increase of sales during December (Figure 5), which makes sense because Christmas and New years represent a season of the year with more transactions in other retail businesses. Furthermore, we highlighted February as the lowest sales month (Figure 5).

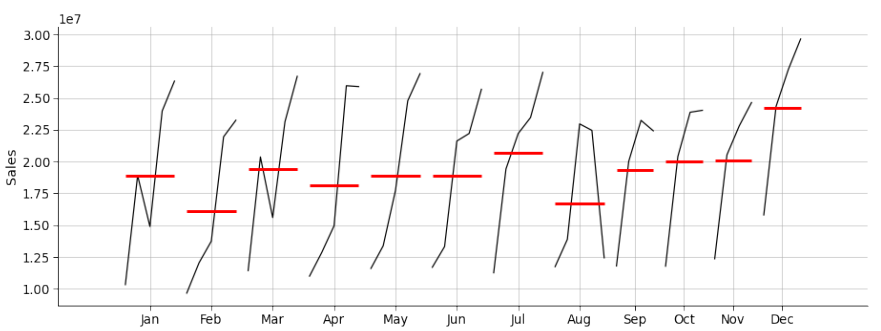

Figure 5: Analysis of seasonality. The y-axis represents the sales aggregated by month, x-axis shows the analyzed months (from January to December), the horizontal red lines denote the sales average by month (55.5 months).

Holidays are one of the most important factors for seasonality, and the Kaggle dataset provided us with relevant holidays and special events in Ecuador (see Table 1). Thus, we analyzed the effects of holidays on sales (see Figure 6). We observed a reasonable correlation between holidays and sales. Consequently, holidays must be incorporated on the forecast.

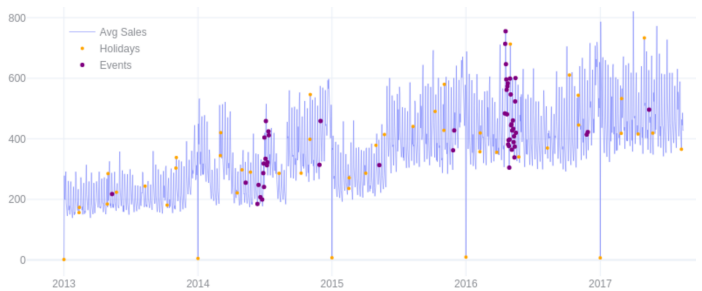

Figure 6: Holidays effect. Holidays and special events are represented as yellow, and purple dots, respectively.

IV

Time-series Forecast

The main goal of the project is to generate a reliable forecast using the sales data from Favorita stores. After the eda, we concluded that our data is not stationarity, possesses a strong seasonal effect, and holidays is a factor that should be considered in the forecast. Therefore, the following models could be implemented:

- **SARIMAX**: Seasonal ARIMA using eXternal data. Further, ARIMA stands for Auto-Regressive (p: dependence on past values), Integrated (d: differencing) Moving Average (q: dependence on past forecast errors).
- **LSTM**: Long-Short Term Memory.
- **GRU**: Gated Recurrent Unit.
- **Facebook Prophet**: is a forecasting procedure based on a general additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.
- **NeuralProphet**: is a hybrid forecasting framework based on PyTorch and trained with standard deep learning methods.

- **LightGBM** (or other extreme gradient boosted method, *e.g.* XGBoost): Light Gradient-Boosting Machine.

Ideally, we would like to implement all of the machine learning models mentioned above, do a benchmark taking into account: performance, computational-resources, training-time, and model-explainability; and compare their results. Nonetheless, due to time-constraints restrictions, we selected **Facebook Prophet** as our unique machine learning model to forecast the sales from Favorita stores.

The reasons to select Facebook Prophet include: a simple and flexible model, see Equation 1, that takes into account multiple human scale seasonality, holidays that occur at irregular intervals, trends that are non-linear, short training time (using the *L-BFGS* optimization algorithm) , low computation-resource demands, a Python module to easily implement the model (under the hood uses Stan), model explainability, and previous experience with the model.

IV.1. Prophet as our Forecast model

Facebook Prophet or Prophet (we are going to use Prophet from now on) is an automated forecasting procedure based on a general additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.

According to Taylor *et al.*, Prophet works best with time series that have strong seasonal effects (which is our case see Figure 4C, and Figure 5), holidays effect (Figure 6), and typically handles outliers well. The additive models contains three main components: **1)** trend, **2)** seasonality, and **3)** holidays, which are combined in Equation 1:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \tag{1}$$

$$g(t) = (\kappa + a(t)^T\delta)t + (m + a(t)^T\gamma) \tag{2}$$

$$s(t) = X(t)\beta \tag{3}$$

$$h(t) = Z(t)\kappa \tag{4}$$

$$\varepsilon_t \sim N(0, \sigma^2) \tag{5}$$

Where $g(t)$ is the trend function which models non-periodic changes in the value of the time series, $s(t)$ represents seasonality (*e.g.*, weekly and yearly seasonality), and $h(t)$ represents the effects of holidays which occur on potentially irregular schedules over one or more days. The error term ε_t represents changes not explained by the model; under the assumption that ε_t is normally distributed.

In Equation 2, κ stands for the growth rate, δ has the rate adjustments, m is the offset parameter, and γ_j is set to $-s_j\delta_j$ to make the function continuous. Next, Equation 3 represents seasonality which can be further extended as:

$$s(t) = \sum_{n=1}^{N} \left(a_n cos\left(\frac{2\pi nt}{P}\right) + b_n sin\left(\frac{2\pi nt}{P}\right)\right)$$

Let P be the regular period we expect the time series to have (*e.g.* $P = 365.25$ and $P = 7$ for yearly and weekly data, respectively). For yearly and weekly seasonality, the authors have found $N = 10$ and $N = 3$ to work well for most series problems, respectively. According to Taylor *et al.*, $\beta \sim Normal(0, \sigma^2)$ to impose a smoothing prior on the seasonality. Finally, Equation 4 which represents the effects of holidays is extended as:

$$Z(t) = [1(t \in D_1), ..., 1(t \in D_L)]$$

By assuming that the effects of holidays are independent. For each holiday i, let D_i be the set of past and future dates for that holiday. And assign each holiday a parameter κ_i which is the corresponding change in the forecast. As with seasonality, we use a prior $\kappa \sim Normal(0, \nu^2)$.

IV.1.1. Prophet Hyperparameters

Prophet possesses 5 hyperparameters which can be tuned through cross validation. The hyperparameters are: **1)** *changepoint prior scale*, **2)** *seasonality prior scale* , **3)** *holidays prior scale* , **4)** *seasonality model* , and **5)** *changepoint range* (this one not recommended to be tuned according to the Prophet developers).

In our work, only the *seasonality mode* (either additive or multiplicative) was tuned using cross validation (CV) with an horizon of 1 year (365 years), period of 180 days, and 730 days for training. Information about CV can be found on the Prophet site under the diagnostics section, or at Taylor *et al.* work.

IV.2. Performance Metric

To assess the performance of our models, we selected the mean absolute percentage error (MAPE) which is defined below:

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

Where A_t is the actual value (true value) and F_t is the forecast result (predicted value). MAPE was selected as a performance metric because of its simplicity to represent the forecast errors within a percentage scale, where the lower the better.

IV.3. Forecast Results

The sales data can be organized by store (54 stores) and by store and family product (1782 combinations). Therefore, as a starting point we aggregate all sales and generate 1 forecast (see General Sales Forecast). Then, generate a forecast by each store (54 forecasts; see Sales Forecast by Store). Finally, we generate a forecast of each product within each store (1782 forecasts; see Sales Forecast by store and product).

IV.3.1. General Sales Forecast

Figure 7 shows the forecast outcome obtained aggregating all the sales, obtaining a MAPE of 22.26% using a half year horizon. After experimenting with different hyperparameters and conditions, the Prophet model with the highest performance (the lowest MAPE) considers the following:

1. A multiplicative seasonality mode.
2. Uses the provided holidays (holiday_events.csv).
3. Custom regressor representing the store closing days.

We observe a correct modeling of the training data, and correctly modeling the pronounced dips around Christmas and New Year (Figure 7). Model explainability is an important factor to select a final model, in addition to performance, and computational training time. Prophet comes with an integrated explanation of the model

components including the effects of holidays, weekly and yearly seasonality which helps to assess the performance of the forecast.

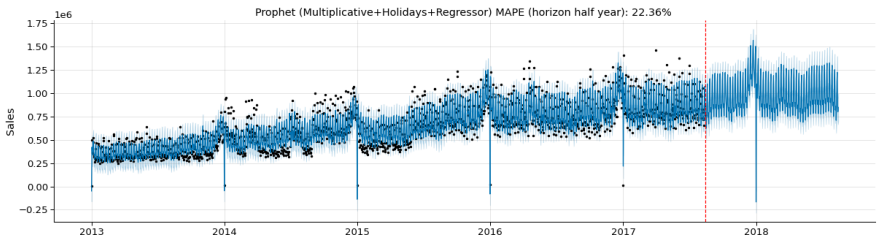

Figure 7: General Sales Forecast. Blue lines denotes the forecast, and black dots describe the actual values. CV was implemented with 730 days leading to 4 forecasts cutoffs. Forecast (half year horizon) is represented after the red dashed line.

In consequence, the forecast components are displayed in Figure 8. A sales increase is reported on the holiday component (Figure 8A), which is expected in the retail business. In terms of weekly and yearly seasonality, Saturday and Sunday; and December are the periods with the highest sales, respectively (Figure 8B and C).

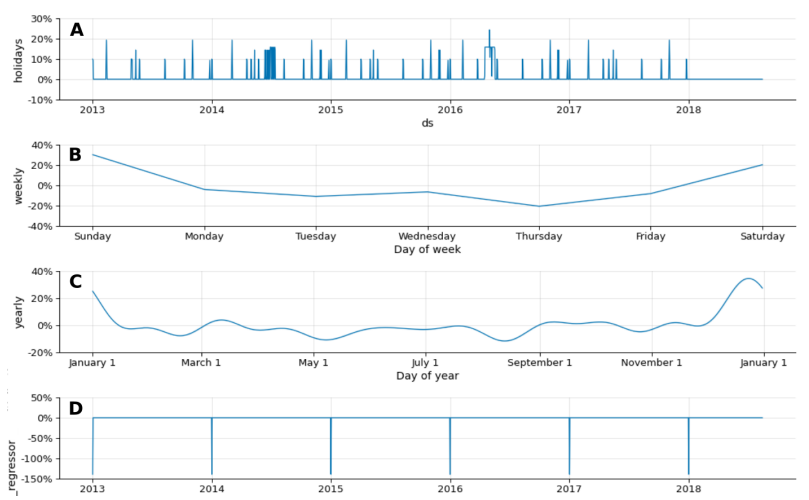

Figure 8: General Forecast Components. The components of the final Prophet model, which considers a multiplicative seasonality, holidays, and a custom regressor represented on Figure 7. **(A)** The holiday component. **(B)** Weekly seasonality. **(C)** Yearly seasonality. **(D)** Regressor considering the store closing days.

The regressor displays the store closing days which explains the models the pronounced dips around Christmas and New Year (Figure 8D). In conclusion, we can state that our proposed models possess a moderate error (MAPE of 22.26%), and the forecast components are in alignment of what we would expect of the retail business.

IV.3.2. Sales Forecast by Store

Our dataset contains 54 stores. Thus, one time series model was produced to each store leading to 54 different Prophet models. In this scenario, a vanilla Prophet model was fitted to each store. Obtaining a mean MAPE of 36.87%, twelve random stores are represented by Figure 9, where the orange lines represent the sales forecast by store. For better figure representation the stores are displayed from 2017-05 to 2017-09 to highlight the last dates.

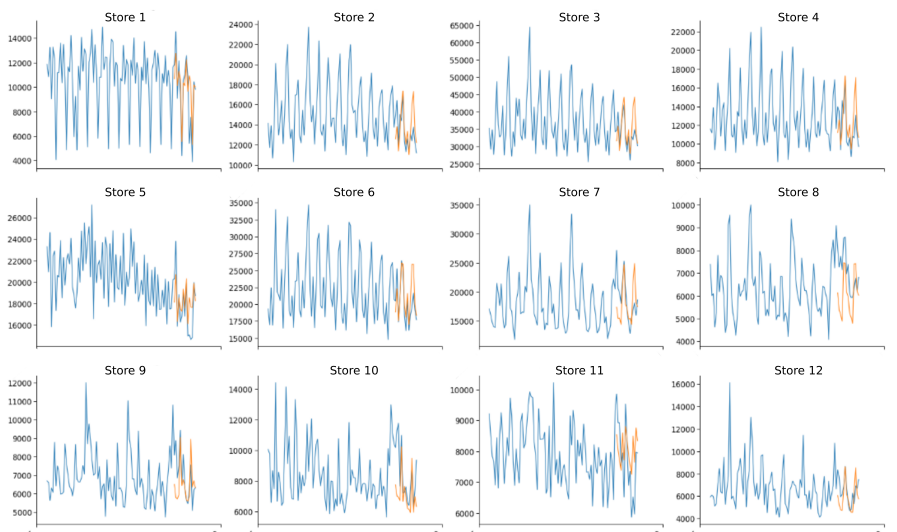

Figure 9: Forecast by Store. The x-axis represents the dates from 2017-05 to 2017-09, the y-axis shows the sales for each store. The blue and orange lines display the store sales and forecast, respectively.

As expected, we obtained a higher error compared to the general sales forecast (36.87% vs. 22.26%). The reasons are: each store has a different sales pattern, and each Prophet model should be tuned for each store. In addition, a data normalization should be implemented to reduce the variance, and achieve a lower MAPE. Nonetheless some store predictions are reasonable, for instance for stores: 1, 3, 6, and 10 (Figure 9). In contrast, for stores: 7, 8, and 11 the forecasts can be considered as

non-correct (Figure 9). Overall, 88% of the forecast can be considered as correct with a mean MAPE of 25.96% (see all the forecast plots at the GitHub repository). Considering the complexity of time series forecasting and the different sales patterns for each store, we can label our results as satisfactory.

IV.3.3. Sales Forecast by store and product

In this scenario, one vanilla Prophet model was fitted for each product for each store producing a total of 1782 forecasts. As we can see on Figure 10, each product for each store presents a completely different sales behavior leading to a more complex forecast problem. On average, we obtained a MAPE of 61.02%. Consequently, we are not able to suggest to rely on our predictions following this approach.

Figure 10: Forecast by Store and Product. The x-axis represents the dates from 2017-05 to 2017-09, the y-axis shows the sales for each store and product. The different color lines represent each sales product.

V

Recommendations

Appendix

VI

Supplementary information

This work was written with emacs[1] using LaTeX[2], using only **Free and Open Source software**. All the computational analysis were carried out using Linux-based distributions. The figures were generated with Python (matplotlib[3]/seaborn[4]/plotly[5]) and Inkscape[6].

Contact: razielar@gmail.com

[1] https://www.gnu.org/software/emacs/
[2] https://www.latex-project.org/
[3] https://matplotlib.org/
[4] https://seaborn.pydata.org/
[5] https://plotly.com/
[6] https://inkscape.org/